M&G No. 50037.30USU1/MS #171320.2

METHOD AND SYSTEM FOR INTERACTING WITH DEVICES HAVING DIFFERENT CAPABILITIES

Related Application

This application is a Utility Patent application based on a previously filed U.S. Provisional Patent application, U.S. Serial No. 60/276,394 filed on March 16, 2001, the benefit of the filing date of which is hereby claimed under 35 U.S.C. § 119(e).

Field of the Invention

The present invention relates generally to computer-executable software, and more particularly to interacting with devices having different capabilities.

Background

With the increasing number and variety of consumer electronics, it is becoming more difficult to write computer applications that can interact with each device without being modified to account for variations between the capabilities of devices. For example, a cell phone may display a list of menu options by placing each option and a number on each display line. Placing a number next to each option facilitates the selection of an option as a cell phone typically includes a numeric keypad. A POCKET PC, on the other hand, usually uses a stylus for inputting user choices and data. The menu options displayed on a POCKET PC may therefore be optimized for selection by a stylus and not include numbers as such a device typically does not have a keypad.

Furthermore, a cell phone display may have a relatively small area in which to display information. As a consequence, menus displayed on a cell phone may be unable to fit on one display screen. Thus, to display a longer menu on a cell phone may require scrolling through information. Conversely, a POCKET PC may have a relatively large area in which to display information. The same menu that required multiple pages for the cell phone to display may only require one page for the POCKET PC to display.

Two different devices may have widely disparate ways of presenting a menu and receiving user input. One device may display the menu using text and receive input through a keypad while another device may "display" the menu through audio and receive input through voice commands.

Writing a program that automatically adapts itself for each device with which it interacts places an undue burden on a programmer. One approach to solving this problem uses an extensible style sheet language (XSL). XSL may be used to transform one extensible markup language (XML) schema to another XML schema. By transforming one schema to another, XSL may be used to translate the commands a program generates for displaying information on a device into commands the display device requires.

Using XSL, however, has several disadvantages. One disadvantage is the number of style sheets required. A new application typically requires one or more new XML schemas specific to that application. Each new XML schema requires at least one new XSL style sheet for each device supported. Thus, to add one application that could be translated to N devices would require N times the number of XML schemas specific to the application. Furthermore, when a new device is created, to provide universal support from all existing applications, new style sheets for each application would be required. This causes a large up-front cost of creating such documents and a lingering cost of maintaining the documents as device features are updated or change.

Another disadvantage of using style sheets is that XSL is not well-suited for two-way interaction with a device. That is, while XSL might be used to translate a particular form onto the display of a device, other mechanisms would be required (and one or more additional style sheets) to translate responses from the device back into information the application could utilize.

Yet another disadvantage of using style sheets relates to maintaining state information regarding a display operation. In the cell phone example above in which a menu requires multiple pages and scrolling to be displayed, XSL is ill-suited for maintaining state information about which page the cell phone is currently on and which page should be sent next to the cell phone.

Thus, there is a need in the art of a method and system for interacting with devices having different capabilities.

Summary

The present invention provides a method and system for interacting with devices having different capabilities. The invention provides intelligent server-side objects (hereinafter referred to as adapters) that translate information and commands to and from various formats depending on the requirements and capabilities of the target device. An interface may be used with the adapters to create interactive forms such that a software developer is able to create a form without knowing the exact details or features of the device upon which the form will be displayed. This allows future use of the form on devices that may not presently have adapters. Additionally, adapters provide a mechanism for developers providing support for new devices to relatively quickly integrate and make compatible with existing server applications new or existing devices.

Brief Description of the Drawings

FIGURE 1 shows an exemplary computing device that may be included in a system implementing the invention;

FIGURE 2 shows a functional block diagram illustrating an exemplary environment for practicing the invention;

FIGURE 3 shows a functional block diagram illustrating an exemplary environment for practicing the invention with another view of some of the components shown in FIGURE 2;

FIGURE 4 shows a functional block diagram of server components transforming information from server objects using adapters and receiving input;

FIGURE 5 shows a multiple dispatch table that may be used by an adapter selector to select an appropriate adapter;

FIGURE 6 shows an extensible document that may be used to define adapters and relationships between adapters;

FIGURE 7 shows a logical flow diagram illustrating a process for selecting an adapter set suitable for use with a device; and

FIGURE 8 shows a logical flow diagram illustrating a process for selecting an adapter to use with an object associated with a server object in accordance with the invention.

Detailed Description

The present invention provides a method and system for interacting with devices having different capabilities. Among other things, disclosed is a system which uses device capabilities to select an appropriate adapter set for interacting with the device. First, an illustrative computing device and operating environment will be described. Then, components used for selecting an adapter will be discussed. Finally, methods for using the computing device and components to select appropriate adapters will be disclosed.

Illustrative Computing Device

FIGURE 1 shows an exemplary computing device that may be included in a system implementing the invention, according to one embodiment of the invention. In a very basic configuration, computing device 100 typically includes at least one processing unit 102 and system memory 104. Processing unit 102 includes existing physical processors, those in design, multiple processors acting together, virtual processors, and any other device or software program capable of interpreting binary executable instructions. Depending on the exact configuration and type of computing device, system memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory 104 typically includes an operating system 105, one or more program modules 106, and may include program data 107. This basic configuration is illustrated in FIGURE 1 by those components within dashed line 108.

Computing device 100 may also have additional features or functionality. For example, computing device 100 may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in Figure 1 by

removable storage 109 and non-removable storage 110. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules or other data. System memory 104, removable storage 109 and non-removable storage 110 are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device 100. Any such computer storage media may be part of device 100. Computing device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 114 such as a display, speakers, printer, etc. may also be included. All these devices are known in the art and need not be discussed at length here.

Computing device 100 may also contain communications connection(s) 116 that allow the device to communicate with other computing devices 118, such as over a network. Communications connection(s) 116 is an example of communication media. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

Illustrative Operating Environment

FIGURE 2 shows a functional block diagram illustrating an exemplary environment for practicing the invention, according to one embodiment of the invention. The environment includes server 200, software development

environment 205, adapter development environment 210, network 215, and mobile devices 220_{a-c}. Server 200 includes device interaction component 400 which is described in more detail in conjunction with FIGURE 4.

Software development environment 205 provides a software developer access for developing applications for server 200. Software development environment 205 may be as simple as a text editor used with a file transport protocol (FTP) application for transmitting and receiving programs from server 200, or it may include a suite of software development tools such as one or more compilers, debuggers, source code control applications, team development tools, and the like. One such suite of software development tools is Microsoft VISUAL STUDIO® produced by Microsoft Corporation of Redmond, Washington. Such software development tools typically make the software applications easier to develop, debug, and maintain, as is understood by those of ordinary skill in the art.

Software development typically involves creating pages, forms, controls, and other server objects (hereinafter sometimes collectively referred to as server objects) for displaying information to and receiving input from users. A server application program may include many such server objects. Typically, the server application program is arranged in terms of pages, i.e. information that should be displayed to a user together. A page may include links to other pages, forms, controls, and other server objects. A form may be used, for example, for collecting address information. The form may display address fields, prompt a user for address and name information, validate inputted information, and send the information to the server application program for further storage and use.

A form may have controls on it to facilitate user input. For example, a form may have a radio button control for receiving a user's selection. A form may have a free text control for receiving textual input from the user. A form may have control buttons such as OK or CANCEL to receive confirmation or cancellation from a user. A control, however, is not limited to being placed within a form; it may also be placed within a page, another control, or another server object.

Adapter development environment 210 provides an adapter developer access to creating adapters for device interaction component 400. In one embodiment,

adapter development environment 210 may be a software development environment similar to software development environment 205. In fact, adapter development environment 210 may be included in software development environment 205. Furthermore, adapter development environment 210 may be distinguishable from software development environment 205 only by the fact that the software developer is writing an adapter rather than other software. In another embodiment, adapter development environment 210 may be a specialized development environment for use in creating adapters. For example, it may have unique menu options, emulation tools, or features that are particularly suited for creating adapters.

Adapters are described in more detail in conjunction with FIGURE 4. Briefly, an adapter's functions include 1) transforming information from server objects into information displayed on an electronic device, such as mobile devices 220_{a-c}; and 2) transforming responses from such electronic devices into information usable by an application program running on the server. For example, an adapter may transform a menu control created using software development environment 205 into a multi-page menu displayed on mobile device 220_a or into a single page menu displayed on mobile device 220_b. The adapter may then transform a menu selection entered by a user into data for use by the application program.

Mobile devices 220_{a-c} include such things as cell phones, pagers, POCKET PCs, hand-held electronic devices, programmable and non-programmable consumer electronics, personal computers, and the like. Such devices typically range widely in terms of capabilities and features. For example, a cell phone may have a numeric keypad and a few lines of monochrome LCD display on which only text may be displayed. A POCKET PC may have a touch sensitive screen, a stylus, and several lines of color LCD display in which both text and graphics may be displayed. A computer may have a keyboard, mouse, speakers, microphone, and a relatively large area on which to display forms.

Network 215 connects device interaction component 400 with mobile devices 220_{a-c}. Network 215 includes wireless and non-wireless networks and networks including a combination of wireless and non-wireless networks. Network 215 may include local area networks (LANs), such as a corporate networking system, wide area

networks, such as the Internet, cellular and/or pager networks, a direct network connection between computers, such as through a universal serial bus (USB) connection, combinations thereof, and the like. In essence, network 215 includes any communication method by which information may travel from any of mobile devices 220_{a-c} to device interaction component 400.

Server 200 is an example of a computing device, such as computing device 100 as described in conjunction with FIGURE 1. Server 200 includes device interaction component 400 which is described in more detail in conjunction with FIGURE 4. Server 200 may also include other application programs and components and may be used for a variety of purposes related or unrelated to the present invention. Server 200 stores, retrieves, and executes applications and/or objects created using software development environment 205. Server 200 stores and retrieves adapters created by adapter development environment 210.

Device interaction component 400 executes on server 200 and utilizes server objects created by software development environment 205 and adapters created by adapter development environment 210. Device interaction component 400 is described in more detail in conjunction with FIGURE 4. Briefly, device interaction component 400 selects appropriate adapters to transform pages, forms, controls, and the like into information suitable for viewing on and receiving user response from mobile devices 220_{a-c}. Device interaction component 400 shields a software developer from the capability intricacies of each mobile device by providing a common interface for use in software development. The interface may allow the developer, for example, to specify that a menu be displayed on a device and that a menu selection be returned. With this interface, the software developer may not need to be concerned with the size of the display or the input capabilities of the user's device as device interaction component 400 transforms the menu in a manner appropriate to the device the user is using. Note, however, that the interface may also provide direct access to the device's capabilities. This allows a software developer to customize interactions with a device if desired.

FIGURE 3 shows a functional block diagram illustrating an exemplary environment for practicing the invention with another view of some of the components

shown in FIGURE 2, according to one embodiment of the invention. FIGURE 3 elaborates on steps a software developer may engage in when creating server objects using software development environment 205. Additionally, FIGURE 3 shows steps device interaction component 400 may execute to display content on mobile devices 220_{a-c}. FIGURE 3 also includes a line indicating an environment in which the server pages, objects, forms, and controls may be developed (development environment) and a line indicating an environment in which the server objects may be utilized to interact with mobile devices 220_{a-c} (production environment).

FIGURE 4 shows a functional block diagram of server components transforming information from server objects using adapters and receiving input, according to one embodiment of the invention. Server 200 includes device interaction component 400 and network interface 450. Device interaction component 400 includes server application objects 405, adapter store 410, device interaction engine 415, adapter selector 420, page adapter 425, device capabilities component 430, form/control adapters 435, receiver 440, and writer 445.

Server application objects 405 store server objects created for a software application. Typically, server objects are not constructed for use on only one device; rather, a developer typically creates the server objects by programming to a specified programming interface. As previously mentioned, the interface abstracts device capabilities such that the developer does not need to know (but can still access if desired) the exact capabilities of the device in order to create an object to display on the device. Instead, the developer may create an object which calls a feature-rich programming interface and relies on adapters (discussed below) to transform such calls in device-specific ways.

Adapter store 410 may store adapters, information as to where adapters may be found, other information about adapters, or any combination thereof. Typically, adapters are arranged in adapter sets. That is, the adapters associated with a particular device or set of devices are grouped (at least logically) in an adapter set. For example, a set of devices may communicate using a wireless markup language (WML). A set of adapters may be logically grouped to handle conversion to and from WML. One adapter may be used to transform a server menu control to display menus and receive

user selections from a device. Another adapter may be used to transform a server free-text question control to display a question and retrieve free-form text from the same device. Another adapter may be used to translate a server radio button control into a format suitable for display on the device and to receive a user's selection. Another adapter may be used to transform a server spreadsheet control to display and receive spreadsheet type data on a device.

Adapters may transform forms such that from a device's perspective (or a user using the device), it is difficult or impossible to determine the exact format of the server object containing the form. For example, a server menu form transformed by an adapter and displayed on a cell phone may list items and numbers next to the items for user input. A radio button control transformed by an adapter and displayed on the cell phone may also list items and numbers next to the items for user input. This may occur because the cell phone lacks a radio button interface. Rather than preclude a software developer from using a radio button form, an adapter may be created that transforms a radio button server object into what appears to the user to be a menu form.

An adapter may inherit attributes and methods from another adapter in the same or another adapter set. An adapter set may inherit adapter associations, i.e., which server objects should be mapped to which adapters, from another adapter set. Methods and attributes of ancestor adapters may be extended, restricted, or over-written. Generally, object-oriented rules apply to the relationships and interactions between related adapters. This model makes it easier to create an adapter set for a new but similar device. For example, one device may accept hypertext markup language (HTML) documents. A new device may accept compact hypertext markup language (CHTML) documents. Causing a CHTML adapter set to inherit from an already-existing HTML adapter set may eliminate a significant amount of work in creating the CHTML adapter set.

Adapter selector 420 receives device capabilities and a server object, e.g. a form, page, or control, from device interaction engine 415 and selects an adapter for transforming the server object. A device may not match with an adapter set. That is, an adapter may not exist in any adapter set for transforming the server object to the device. In that case, a default adapter set may be used, an error may be generated, or other

processing may take place. For example, adapter selector 420 may indicate to device interaction engine 415 that no adapter set matches the device capabilities. Typically, adapter selector 420 sends device interaction engine 415 the selected adapter, a reference to it, or an error. Adapter selector 420 may also be used to determine which adapter set should be used to map server objects to a device. Selection of adapters is discussed in more detail in conjunction with FIGURES 5-8.

Receiver 440 receives requests, responses, and/or information from network interface 450. In one embodiment, such requests, responses, and/or information are sent directly to receiver 440. In another embodiment, such requests, responses, and/or information are sent to device interaction component 415 and relayed to form/control adapters 435. Typically, when a communication is a response to a previous communication sent by form/control adapters 435, the response is relayed to form/control adapters 435 for further processing. For example, if a device is responding to a menu selection sent by an adapter, the response may be relayed to form/control adapters 435 for further processing. When a communication is a request for access to a server object, receiver 440 may request device capabilities from device capabilities component 430 and send these capabilities together with the communication to device interaction engine 415.

In another embodiment of the invention, when the communication is a request, the communication may be sent directly to device interaction engine 415 which then requests device capabilities from device capabilities component 430. In such embodiments, device capabilities component 430 may be directly connected to or under control of device interaction engine 415 in addition to, or in lieu of, being connected to or under control of receiver 440.

Device capabilities component 430 determines what capabilities a device has. Different devices may have different capabilities as discussed in conjunction with FIGURE 2. Device capabilities component 430 may include a database of "known" devices or it may query a device on-the-fly for capabilities. Device capabilities component 430 may determine that the device capabilities are unknown. In such a case, device capabilities component 430 may send a default set of capabilities, an error, or some other message so indicating.

Writer 445 sends information to network interface 450 directed at one or more devices. Although form/control adapters 435 are shown directly connecting to writer 445, writer 445 may receive information from any adapters including page adapter 425, form/control adapters 435, and/or any other adapters. In one embodiment of the invention, writer 445 may be implemented as an object having certain methods, helper functions, and attributes. Writer 445 may be passed to each adapter performing a transformation. Each adapter performing a transformation uses writer 445 to insert information into a response to be sent to a device.

Page adapter 425 may be instantiated by device interaction engine 415 or by an executing server object spawned by device interaction engine 415. As the server object executes, it may request that a page be rendered or that information be requested from a user using a device. Upon request (through invocation of one of page adapter 425's methods), page adapter 425 begins rendering a "page" of information to deliver to the device together with any controls necessary to process the server object's request.

In one embodiment of the invention, once a server object from server application objects 405 begins execution and page adapter 425 is instantiated, page adapter 425 may cease communicating with device interaction engine 415. Instead, it may receive commands from and deliver information to the associated executing server object. In other words, device interaction engine 415 may instantiate page adapter 425, execute an appropriate server object from server application objects 405, associate the instantiated page adapter 425 with the executing server object, and "step out of the way" as the executing server object and adapter interact with each other to send information to and receive information from a device. Device interaction engine 415 may then be available to service a request from another device by executing another instance of the same or a different server object, instantiating another page adapter, and associating the server object with the new page adapter.

In another embodiment of the invention, device interaction engine 415 may be more involved. It may perform tasks such as instantiating page adapter 425, executing an appropriate server object from server application objects 405, associating the instantiated page adapter with the executing server object, instantiating and

associating one or more form control adapters 435 as needed, relaying requests and/or information between the instantiated adapter(s) and associated executing server object(s), and relaying messages from receiver 440 to adapters(s) and/or server objects as appropriate. In this embodiment of the invention, device interaction engine 415 may receive communications, determine what should be done, and "farm out" work and messages as needed.

In one embodiment of the invention, page adapter 425 may communicate with device interaction engine 415 to determine which form and/or control adapters to instantiate for objects referenced from the server object with which page adapter 425 is associated. In another embodiment of the invention, the associated server object may instantiate the appropriate form and/or control adapters and associate them with page adapter 425 and/or forms and controls referenced within the server object. In yet another embodiment of the invention, device interaction engine 415 uses adapter selector 420 to pre-determine which form and/or control adapters may be needed, instantiates such adapters, and associates them with appropriate server objects.

Form/control adapters interact with page adapter 425, receiver 440, and writer 445, and may interact with each other. Form/control adapters 435 may receive information from and transmit information to page adapter 425. For example, upon receiving instructions to render a page, page adapter may send instructions to each form and/or control adapter associated with the page to render its respective form or control. Form/control adapters 435 may receive information from and transmit information to associated server objects executed from server application objects 405. For example, a server object may instruct one or more forms and/or controls to render themselves without causing that all forms and/or controls render themselves. Form/control adapters 435 may receive information from and transmit information to receiver 440 and writer 445. Form/control adapters 435 may receive information from and transmit information to each other as explained below.

Although FIGURE 4 shows form/control adapters 435 interacting directly with receiver 440 and writer 445, there may be one or more layers of adapters between form/control adapters 435 and receiver 440 and writer 445. For example, a panel form may include a radio button control and a spreadsheet form. Even if one or

more layers of adapters are between form/control adapters 435 and receiver 440 and writer 445, form/control adapters 435 may still communicate directly with receiver 440 and writer 445 and not be limited to communicating requests and information through a sub form or control. On the other hand, forms and controls may communicate with or through each other (even if they are unrelated) to transform information from server objects into information appropriate for receiver 440.

Device interaction engine 415 performs many functions, some of which have been alluded to above. Some of device interaction engine 415's basic functions include receiving a request, selecting and instantiating adapters, and executing appropriate server objects. Device interaction engine 415 may receive a request from receiver 440 indicating that a device is requesting access to one or more server objects contained in server application objects 405. Device interaction engine 415 determines which server object(s) from server application objects 405 should be executed to service the request and executes the determined object(s). Additionally, device interaction engine 415 may employ adapter selector 420 to select appropriate adapter(s) for the server object(s), instantiate the selected adapter(s), and associate the adapter(s) with the server objects(s). Device interaction engine 415 may also be used to communicate requests for adapters to adapter selector 420. For example, an executing server object or an instantiated adapter may request an adapter using device interaction engine 415.

Network interface 450 transmits and receives messages over network 215. Such messages may be transmitted and received using protocols including hypertext transport protocol (HTTP), transmission control protocol/Internet protocol (TCP/IP), ftp, email, direct file transfer, combinations thereof, and the like. In essence any transmission protocol capable for transmitting information over network 215 may be used in conjunction with network interface 450 to send information to and receive information from devices.

Some embodiments of device interaction component 400 and its components have been described above. In light of this disclosure, it will be understood that components and interactions of the components within device interaction component 400 could be changed, added, or removed without departing from the spirit

and scope of this invention. Following is a description of a table that might be stored in adapter store 410 and utilized by adapter selector 420 to select appropriate adapters.

Illustrative Adapter Selection Components

FIGURE 5 shows a multiple dispatch table that may be used by an adapter selector, such as adapter selector 420, to select an appropriate adapter, according to one embodiment of the invention. Along the X axis are displayed labels of forms and controls. Along the Y axis are displayed names of adapter sets. Cells formed from intersecting columns and rows may contain references, sometimes referred to as pointers, to an adapter that performs transformation for the adapter set for the column-indicated form or control.

Multiple dispatching allows the selection of the code to execute to be based upon the subtypes of more than one argument whereas single dispatching selects code to execute based on the subtype of one argument. Polymorphism is one example of single dispatching. With polymorphism, one typically defines a class with a virtual method. Then, one defines one or more child classes that inherit from the class. Typically, the child classes will each define a method that is called when the virtual method would have been called. In a classic example, the child classes operate on shapes such as squares or circles. The virtual method is called with the subtype of child class (one argument), which then causes the appropriate code to be executed to draw the shape, e.g., circle drawing or square drawing code, depending on the child's class type.

Double dispatching allows the code selected to be based upon the subtypes of two arguments. For example, the drawing code could be selected based on subtypes of sphere and wire frame to draw a sphere using a wire frame. Double dispatching may also be referred to as multiple dispatching of degree two. Multiple dispatching has several advantages known in the art over single dispatching. While the table shown in FIGURE 5 may be used for double dispatching, it might also be extended in multiple dimensions to provide for more degrees of dispatching.

As described earlier in conjunction with FIGURE 4, adapter sets may inherit from other adapter sets. For example, the adapter set CHTML may inherit from the adapter set HTML. In addition server objects, e.g., forms, controls, and pages, may

inherit from other server objects. For example, a RangeValidator control that insures input is within a given range may inherit from a BaseValidator control.

Selecting an appropriate adapter using multiple dispatching is described in more detail in conjunction with FIGURE 7 and 8.

FIGURE 6 shows an extensible document that may be used to define adapters and relationships between adapters, according to one embodiment of the invention. In the document, a tag of "device" (601-603) indicates that an adapter set follows. A tag of "inheritsFrom" (605) indicates that an adapter set inherits from another adapter set (610). The tags "predicateClass" (615) and "predicateMethod" (620) may be used to select an appropriate adapter set for a particular device. For example, the combination of these two tags may reference a function that receives information about the requesting device, e.g., the device's capabilities, and makes a determination as to whether an adapter set is suitable to working with the device. The tag "control" (625) may be used to associate a control with an adapter.

It will be recognized that the document shown in FIGURE 6 may be readily translated into a table similar to that shown in FIGURE 5. In other words, the document could be used to provide a textual interface for multiple dispatching. In light of this disclosure, it will also be recognized that the document may be readily modified to provide for adapters for new devices. For example, to add a new device supporting voice input, a vendor could create an appropriate adapter set. Then, the vendor could edit the document shown in FIGURE 6 and insert a new <device>...</device> section to cause the adapter set to be used in the double dispatching mechanism described in conjunction with FIGURES 5, 7, and 8.

Above have been disclosed an illustrative computing device, an illustrative operating environment, details of a device interaction component, and an exemplary table used in selecting adapters and adapter sets (through multiple dispatching). In addition an exemplary document has been described which simplifies device addition by specifying relationships between adapters and adapter sets as well as where adapters are implemented. Following are disclosed exemplary methods of selecting an appropriate adapter set, page adapter, and form or control adapter.

Illustrative Adapter Selection Components

FIGURE 7 shows a logical flow diagram illustrating a process for selecting an adapter set suitable for use with a device. The process begins at block 705 when a device, such as mobile device 220_a of FIGURE 2, requests access to a server object, such as a form object in server application objects 405 of device interaction component 400 of FIGURE 4.

At block 710, the device capabilities are determined. For example, referring to FIGURE 4, receiver 440 requests device capabilities from device capabilities component 430. Then receiver 440 sends the request from mobile device 220_a together with the device's capabilities to device interaction engine 415.

At block 715, a loop is entered to determine an adapter set appropriate for interacting with the device. Each time the loop iterates, a determination is made as to whether another adapter set is available for consideration for interacting with the device. If an adapter set is not available, processing branches to block 735. Otherwise, processing branches to block 720. Continuing with the example above, device interaction engine 415 requests that adapter selector 420 determine an appropriate adapter set for mobile device 220_a. Adapter selector 420 begins searching through adapter sets in adapter store 410, to find an appropriate adapter set.

At block 720, information is retrieved about the adapter set to be considered. Continuing with the example above, adapter selector 420 retrieves an adapter set from adapter store 410. Typically, the set would contain information such as that found in FIGURE 6.

At block 725, a determination is made as to whether the adapter set is applicable to the device capabilities. If the adapter set is applicable to the device capabilities, processing branches to block 730; otherwise, processing branches to block 715. Continuing with the example above, adapter selector 420 uses information from the device capabilities, such as, for example, that mobile device 220_a communicate using WML, and determines if the adapter set works with WML.

At block 730, the page adapter associated with the adapter set is instantiated. If the process executes block 730, this indicates that a suitable adapter set has been located. Continuing with the example above, adapter selector 420 sends a

reference to the WML adapter set to device interaction engine 415. Using the reference, device interaction engine 415 instantiates the page adapter of the WML adapter set, associates the WML adapter set with a server object in server application objects 405, and instantiates and executes the server object.

At block 735, processing ends. At this point, either an adapter set applicable to the device has been located and a page adapter has been instantiated, or an adapter set has not been located that is applicable to the device. In the former case, the server object associated with the page adapter may begin transmitting information to and receiving information from the device using the page adapter and other adapters from the page adapter's adapter set. In the latter case, a default adapter set may be selected and its page adapter instantiated, the device may be sent an error message, or other action may be taken.

FIGURE 8 shows a logical flow diagram illustrating a process for selecting an adapter to use with an object associated with a server object. For example, as mentioned earlier, a page may include a control object, such as a radio button control, and while the page may have an associated page adapter, the control object may still need its own adapter. The object associated with the server object will be referred to in this discussion with the phrase "form or control," although the associated object is not limited to these particular objects.

Briefly, the search for a suitable adapter begins after a page adapter has been selected and a form or control adapter is requested. The search for a suitable adapter for the form or control starts by considering the adapter set from which the page adapter was chosen. If when considering this adapter set, there is not a suitable adapter for the form or control, an ancestor of the form or control is considered with the same adapter set. If a suitable adapter is not found for the ancestor of the form or control, another ancestor of the form or control is selected. After all ancestors of the form or control are considered with the adapter set without finding a suitable adapter, an ancestor adapter set of the adapter set is selected. Again, the original form or control is considered in selecting an appropriate adapter in the selected ancestor adapter set. Then, ancestors of the form or control are selected and considered. This process continues until the form or control and its ancestors have been compared against the

original adapter set and its ancestors. When a suitable adapter is located, the selection process terminates and the adapter is instantiated and associated with the form or control.

The process begins at block 805 when a mapping of a form or control to a device is requested. For example, referring to FIGURE 2, a mobile device, such as mobile device 220_b, may request access to a server object. In response, device interaction engine 415 of FIGURE 4 may instantiate and execute a server object from server application objects 405 and instantiate and associate a page adapter, such as page adapter 425, with the server object. The server object, device interaction engine 415, and/or page adapter 425 may require additional form/control adapters to transform the form or control to a form suitable for the device. For illustrative purposes, assume that the associated object is a radio button control.

At block 810, the form or control and the adapter set of the page adapter previously chosen are selected. Continuing with the example above, a request is sent to adapter selector 420 to find a suitable form/control adapter for use with the form or control. Adapter selector 420 begins by considering the radio button control and the page adapter's adapter set.

At block 815, a search is made for a suitable adapter for the form or control. The search may be made by a table lookup using the name of the form or control. If a suitable adapter is found, processing branches to block 840; otherwise, processing branches to block 820. Continuing with the example above, adapter selector 420 determines if there is a suitable adapter in the adapter set for transforming the radio button control.

At block 820, a determination is made as to whether the form or control has another ancestor that has not been considered with the currently-selected adapter set. If so, processing branches to block 825; otherwise, processing branches to block 830. Continuing with the example above, adapter selector 420 determines whether the radio button control has another ancestor that has not been considered with the currently-selected adapter set.

At block 825, an ancestor of the form or control is selected for further consideration. This ancestor is used in the next iteration of searching for a suitable

adapter. Continuing with the example above, adapter selector 420 selects another ancestor of the form or control that has not been considered with the currently-selected adapter set.

At block 830, a determination is made as to whether the adapter set under consideration inherits from another adapter set. Block 830 is reached after all ancestors of the form or control have been exhausted without finding an appropriate adapter for mapping the form or control. If the adapter set inherits from another adapter set, processing branches to block 835; otherwise, processing branches to block 845. Continuing with the example above, adapter selector 420 determines whether the currently considered adapter set inherits from another adapter set.

At block 835, the original form or control is selected together with an adapter set that is an ancestor of the currently-selected adapter set. This ancestor adapter set is used with the next iteration of searching for an appropriate adapter for the form or control. Continuing with the example above, adapter selector 420 selects an ancestor adapter set and the radio button control.

At block 840, a form/control adapter is instantiated and associated with the form or control. Block 840 is reached when an adapter or one of its ancestor classes is suitable to use with the form or control. Although not shown, block 840 may also be reached if no suitable adapter is found and a default adapter is selected. Continuing with the example above, adapter selector 420 sends the selected adapter to device interaction engine 415 (or another requesting device) which then instantiates the adapter (into one of form/control adapters 435) and associates it with other appropriate adapters and/or the appropriate form or control associated with the server object for which mapping was sought.

At block 845, the process ends. At this point an appropriate or default adapter has been found and instantiated or no adapter has been found which maps to the form or control. When an adapter has been found, the adapter has been associated with other adapters (as appropriate) and with the form or control in the server object for which mapping was sought.

In one embodiment of the invention, when an appropriate adapter has been found for the form or control and no entry exists in a look up table used to find the

adapter, an entry is placed in the table. This speeds future requests to find the appropriate adapter for the particular form or control.

In some embodiments of the invention, selecting ancestor classes of a form or control proceeds in a linear fashion. That is, first the immediate ancestor of the form or control is selected, for example the form or control's parent. Then, the next most immediate ancestor of the form or control is selected, for example, the form's grandparent, etc. In other embodiments of the invention, selecting ancestor classes of a form or control proceeds in other fashions. For example, the most distant ancestor, e.g. a base class, may be selected first, etc. Likewise, in some embodiments of the invention, selecting ancestor adapter sets of an adapter set proceeds in a linear fashion. In other embodiments of the invention, selecting ancestor adapter sets proceeds in other fashions.

The various embodiments of the invention may be implemented as a sequence of computer implemented steps or program modules running on a computing system and/or as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. In light of this disclosure, it will be recognized by one skilled in the art that the functions and operation of the various embodiments disclosed may be implemented in software, in firmware, in special purpose digital logic, or any combination thereof without deviating from the spirit and scope of the present invention as recited within the claims attached hereto.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.